Assignment 4 — Small Worlds

Please complete the exercises listed below and submit your notebook as an `.ipynb` or `PDF` file.

Name: `Anushree Kolhe`

The `socfb-Northwestern25` (https://github.com/CambridgeUniversityPress/FirstCourseNetworkScience/blob/master/datasets/socfb-Northwestern25/socfb-Northwestern25.edges.gz) network in the book's Github repository is a snapshot of Northwestern University's Facebook network. The nodes are anonymous users and the links are friend relationships. Load this network into a NetworkX graph in order to answer the following questions. Be sure to use the proper graph class for an undirected, unweighted network.

In [1]:
```
import pandas as pd
import os
import networkx as nx
```

Load the network (https://github.com/CambridgeUniversityPress/FirstCourseNetworkScience/blob/master/datasets/socfb-Northwestern25/socfb-Northwestern25.edges.gz) using the `networkx.read_edgelist()` function.

In [2]:
```
current_directory = os.getcwd()
# move back a directory
parent_dir = os.path.dirname(current_directory)
G = nx.read_edgelist(parent_dir + '/datasets/socfb-Northwestern25/socfb-Northwestern25.edges.gz')
```

In [3]:
```
nx.info(G)
```

```
/var/folders/_3/98b473cn1k993m8ln5gwyv6h0000gn/T/ipykernel_70649/1064119803.py:1: DeprecationWarning: info is deprecated and will be removed in version 3.0.

  nx.info(G)
```

Out[3]:
```
'Graph with 10567 nodes and 488337 edges'
```

Q1. How many nodes and links are in this network?

In [4]:
```
total_nodes = G.number_of_nodes()
total_nodes
```

Out[4]:
```
10567
```

In [5]:
```
total_edges = G.number_of_edges()
total_edges
```

Out[5]:
```
488337
```

Q2. Which of the following best describes the connectedness of this network?

1. Strongly connected
2. Weakly connected
3. Connected
4. Disconnected

In [6]:
```
1 nx.is_strongly_connected(G)
```

```
---------------------------------------------------------------------------
NetworkXNotImplemented                    Traceback (most recent call last)
Input In [6], in <cell line: 1>()
----> 1 nx.is_strongly_connected(G)

File ~/opt/anaconda3/lib/python3.9/site-packages/networkx/utils/decorators.py:816, in argmap.__call__.<locals
>.func(_argmap__wrapper, *args, **kwargs)
    815 def func(*args, __wrapper=None, **kwargs):
--> 816     return argmap._lazy_compile(__wrapper)(*args, **kwargs)

File <class 'networkx.utils.decorators.argmap'> compilation 13:3, in argmap_is_strongly_connected_10(G)
      1 from collections import defaultdict
      2 from os.path import splitext
----> 3 from contextlib import contextmanager
      4 from pathlib import Path
      6 import networkx as nx

File ~/opt/anaconda3/lib/python3.9/site-packages/networkx/utils/decorators.py:86, in not_implemented_for.<loc
als>._not_implemented_for(g)
     82 def _not_implemented_for(g):
     83     if (mval is None or mval == g.is_multigraph()) and (
     84         dval is None or dval == g.is_directed()
     85     ):
---> 86         raise nx.NetworkXNotImplemented(errmsg)
     88     return g

NetworkXNotImplemented: not implemented for undirected type
```

In [7]:
```
1 nx.is_weakly_connected(G)
```

```
---------------------------------------------------------------------------
NetworkXNotImplemented                    Traceback (most recent call last)
Input In [7], in <cell line: 1>()
----> 1 nx.is_weakly_connected(G)

File ~/opt/anaconda3/lib/python3.9/site-packages/networkx/utils/decorators.py:816, in argmap.__call__.<locals
>.func(_argmap__wrapper, *args, **kwargs)
    815 def func(*args, __wrapper=None, **kwargs):
--> 816     return argmap._lazy_compile(__wrapper)(*args, **kwargs)

File <class 'networkx.utils.decorators.argmap'> compilation 17:3, in argmap_is_weakly_connected_14(G)
      1 from collections import defaultdict
      2 from os.path import splitext
----> 3 from contextlib import contextmanager
      4 from pathlib import Path
      6 import networkx as nx

File ~/opt/anaconda3/lib/python3.9/site-packages/networkx/utils/decorators.py:86, in not_implemented_for.<loc
als>._not_implemented_for(g)
     82 def _not_implemented_for(g):
     83     if (mval is None or mval == g.is_multigraph()) and (
     84         dval is None or dval == g.is_directed()
     85     ):
---> 86         raise nx.NetworkXNotImplemented(errmsg)
     88     return g

NetworkXNotImplemented: not implemented for undirected type
```

In [8]:
```
1 nx.is_connected(G)
```

Out[8]: False

The given network is best described as a Disconnected network.

Q3. Average path lengths.

We want to obtain some idea about the average length of paths in this network, but with large networks like this it is often too computationally expensive to calculate the shortest path between every pair of nodes. If we wanted to compute the shortest path between every pair of nodes in this network, how many shortest-path calculations would be required? In other words, how many pairs of nodes are there in this network? (Hint: Remember this network is undirected and we usually ignore self loops, especially when computing paths.)

In [9]:
```
num_pairs = total_nodes * (total_nodes - 1) // 2
```

In [10]:
```
num_pairs
```

Out[10]:
```
55825461
```

Q4. Sampling for efficiency.

To save time, let's try a sampling approach. You can obtain a random pair of nodes with

```
random.sample(G.nodes, 2)
```

Since this sampling is done without replacement, it prevents you from picking the same node twice. Do this 1000 times and for each such pair of nodes, record the length of the shortest path between them. Take the mean of this sample to obtain an estimate for the average path length in this network. Report your estimate to one decimal place.

In [11]:
```
import random
```

In [12]:
```
sh_path = []

for i in range(1000):
    n1,n2 = random.sample(G.nodes, 2)
    try:
        shortest_path_length = nx.shortest_path_length(G, n1, n2)
        sh_path.append(shortest_path_length)
    except nx.NetworkXNoPath:
        pass
```

```
/var/folders/_3/98b473cn1k993m8ln5gwyv6h0000gn/T/ipykernel_70649/1769285383.py:4: DeprecationWarning: Samplin
g from a set deprecated
since Python 3.9 and will be removed in a subsequent version.
  n1,n2 = random.sample(G.nodes, 2)
```

In [13]:
```
mean_sh_path = sum(sh_path) / len(sh_path)
print("Estimated:", round(mean_sh_path, 1))
```

```
Estimated: 2.7
```

Q5. Apply a slight modification to the above procedure to estimate the diameter of the network. Report the approximate diameter.

In [14]:
```
max_sh_path = 0

for i in range(1000):
    n1,n2 = random.sample(G.nodes, 2)
    try:
        sh_path_length = nx.shortest_path_length(G, n1, n2)
        if shortest_path_length > max_sh_path:
            max_sh_path = sh_path_length
    except nx.NetworkXNoPath:
        pass
```

```
/var/folders/_3/98b473cn1k993m8ln5gwyv6h0000gn/T/ipykernel_70649/1718690404.py:4: DeprecationWarning: Samplin
g from a set deprecated
since Python 3.9 and will be removed in a subsequent version.
  n1,n2 = random.sample(G.nodes, 2)
```

In [15]:
```
print("Diameter:", round(max_sh_path,1))
```

```
Diameter: 3
```

Q6. What is the average clustering coefficient for this network? Answer to at least two decimal places.

In [16]:
```
round(nx.average_clustering(G), 2)
```

Out[16]:
```
0.24
```

Q7. Is this network assortative or disassortative? Answer this question using the two methods shown in the textbook and class. Do the answers differ?

In [17]:
```
r = nx.degree_assortativity_coefficient(G)
r
```

Out[17]:
```
0.03444129080711028
```

In [18]:
```
neighbor_degree_corr = nx.degree_pearson_correlation_coefficient(G)
```

In [20]:
```
print("Degree Assortativity Coefficient:", r)
print("Average Nearest Neighbor Degree Correlation Coefficient:", neighbor_degree_corr)
```

```
Degree Assortativity Coefficient: 0.03444129080711028
Average Nearest Neighbor Degree Correlation Coefficient: 0.034441290807109316
```

Here the assortativity coefficient is positive and thus, we can say that the network is assortative.

In [ ]:
```

```